J.D. Lohn, "Experiments on Evolving Software Models of Analog Circuits,"
Communications of the ACM, Special issue on evolvable hardware, April 1999, pp. 67-69.

Experiments on Evolving Software Models of Analog Circuits

Jason D. Lohn

Analog circuits are of great importance in electronic system design since the world is fundamentally analog in nature. While the amount of digital design activity far outpaces that of analog design, most digital systems require analog modules for interfacing with the external world. It was recently estimated that approximately 60% of digital application-specific integrated circuit designs incorporated analog circuits. With challenging analog circuit design problems and few analog design engineers, there are economic reasons for automating the analog design process, especially time-to-market considerations.

Techniques for analog circuit design automation began appearing about two decades ago. These methods incorporated heuristics [6], knowledge bases [1], simulated annealing [5], and other algorithms. Efforts using techniques from evolutionary computation began appearing over the last few years. These include the use of genetic algorithms to select electronic component values (for example, the resistance value of a resistor), to select circuit topologies, and to design amplifiers using a limited set of canned topologies [4]. A genetic programming-based analog circuit design system has been demonstrated in which the circuit sizes, component values, and the circuit topologies are determined automatically [3]. The genetic-algorithm systems typically represent circuit structures as vectors of parameters encoded in binary strings, while the genetic programming system manipulates tree data structures.

Because evolutionary search is well-known to be sensitive to how candidate solutions are encoded, we devised and experimented with a new encoding system that consists of a sequence of circuit-constructing instructions. Like the "turtle" in the language Logo that can be commanded to draw an image, our language uses an automaton programmed to construct an electrical circuit. The automaton is called a circuit-constructing robot or cc-bot, and the language that programs it is very small and, in its current incarnation, contains only component-placing instructions. The cc-bot language has the desirable property that virtually all possible sequences of instructions result in a valid electrical circuit. This property is important because it greatly limits the "out-of-bounds" regions of the search space containing "illegal" circuits. Thus, an evolutionary algorithm will spend nearly all its time generating valid electrical circuits. While this is a considerable achievement, the ability to generate every possible circuit topology is lost. This is not considered a drawback for the circuit types we investigated since a vast number of topologies and existing circuit designs could be encoded using the cc-bot approach.

To construct a circuit, a template circuit is specified that consists of a power source and an output terminal. As shown in Figure 1, the evolved circuit grows within the dashed box.

Given a list of instructions, the cc-bot begins at the start node and constructs a circuit by placing components sequentially until the last one in the list is placed. If this last component does not connect to the output terminal, by convention it is forced to connect there.

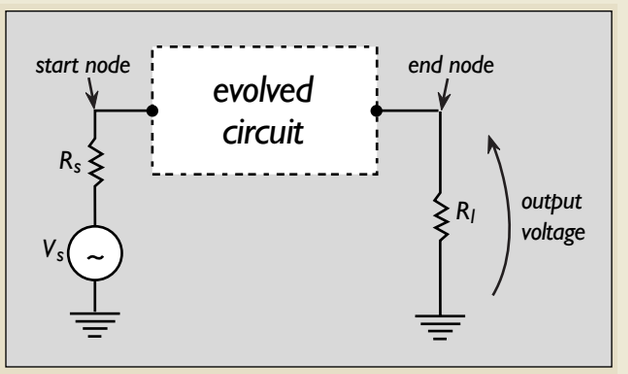

Figure 1. Template circuit: the evolved circuit is located between fixed input and output terminals.

For each component type there are five instructions: move-to-new, cast-to-previous, cast-to-ground, cast-to-input, cast-to-output. If it is desirable to use only resistors and capacitors, for example, then 10 instructions are required: five for resistors and five for capacitors. Each instruction takes a component value (for example, 12.5 ohms) as its sole argument. The move-to-new instruction places one end of a component at the active node and the other at a newly created node (the

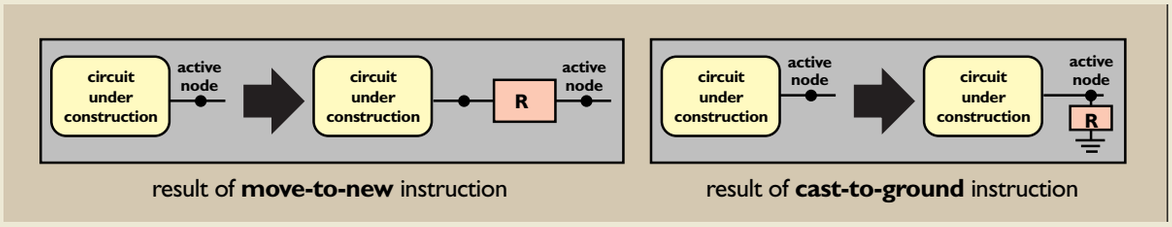

Figure 2. Effect of placing a resistor with move-to-new and cast-to-ground instructions.

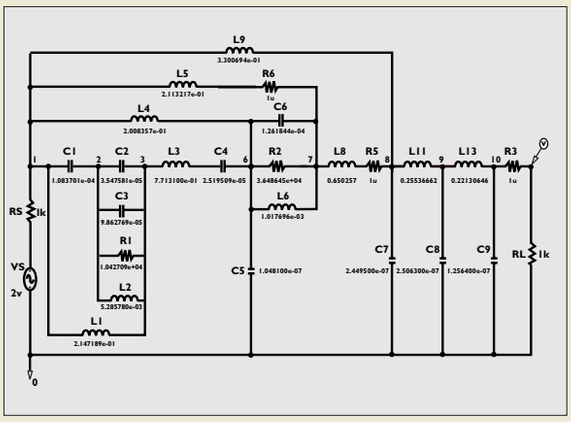

Figure 3. Evolved low-pass filter design.

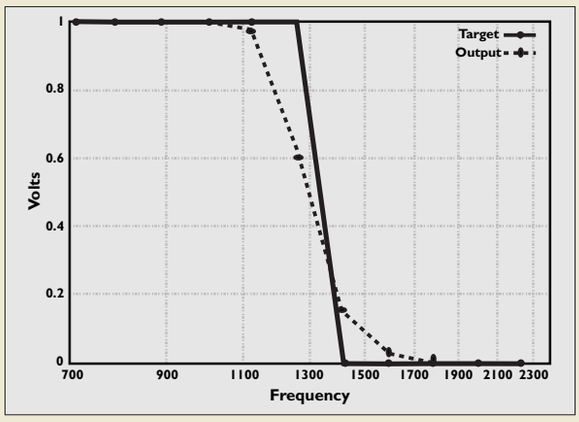

Figure 4. Frequency response of evolved low-pass filter.

"active" node is the current location of the cc-bot). The newly created node then becomes the active node. The cast-to instructions place one end of the component at the active node and the other at either the ground, input, output, or previously created node. After executing a cast-to instruction, the cc-bot remains at its current location. Illustrations of two instructions that place resistors are shown in Figure 2.

Using a hybrid of a standard genetic algorithm and a genetic program, we performed three experiments in which the goals were to evolve low-pass filters. A low-pass filter is a circuit that allows low frequencies to pass through it, but stops high frequencies from doing so. Below the "passband" frequency the input signal is passed to the output, potentially reduced (attenuated) by a specified number of decibels (dB). Above the "stopband" frequency, the input signal is markedly decreased. Between the passband and stopband the frequency response curve transitions from low to high attenuation. Filter design is a well-understood discipline within circuit design. Its "design space" has been widely explored [2], which allows us to compare our evolved designs to existing designs.

In general, evolutionary algorithms are well-suited for parallelization. In our system, circuit simulations run as parallel processes on a network of workstations. A host computer maintains the population of individuals and distributes them to worker nodes using socket connections. The worker nodes decode the individuals into representations suitable for the circuit simulator. The circuits are then simulated and fitness values are calculated. Hundreds of individuals and fitness scores are packaged into a single message so that network congestion delays are minimized.

The design tasks posed in our three experiments increased in difficulty, starting first with a low-pass filter found in an electronic stethoscope design. Our system found a suitable circuit after evaluating approximately 10,000 circuits. The goal of the second experiment was to find a filter with more stringent specifications called a 3rd-order Butterworth filter. A circuit was found during a run that evaluated roughly 400,000 circuits and lasted approximately four hours using six workstations.

The third experiment had the most stringent

specifications. We required the passband to have a very small attenuation of 0.01dB, the stopband to have a very high attenuation of 60dB, and the transition to occur between 1kHz and 2kHz. In other words, the output signal should be roughly 99.8% of the input in the passband, and 1000 times smaller than the input in the stopband. After running for six hours on five workstations, and completing roughly 930,000 circuit evaluations, an evolved circuit that met the specifications was found. The evolved circuit and its frequency response are shown in Figures 3 and 4.

Our experimental results using a new circuit-constructing language were encouraging. Although our technique is topology-limited, the ability of our system to produce useful circuits was demonstrated. Recently we extended our circuit-constructing language to incorporate transistors. The initial experimental results were encouraging: amplifiers, which are very common in analog design, having high gain values were found in under six million circuit evaluations. With the previous successes in evolving analog circuits, and the encouraging results of our system, we are optimistic that a subset of analog circuit design tasks may be routinely accomplished by means of evolutionary computation in the future. C

References

1. Harjani, R., Rutenbar, R.A., and Carey, L.R. A prototype framework for knowledge-based analog circuit synthesis. In *Proceedings of the 24th Design Automation Conference*, 1987.
2. Huelsman, L.P. *Active and Passive Analog Filter Design.* McGraw-Hill, NY, 1993.
3. Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A., and Dunlap, F. Automated synthesis of analog electrical circuits by means of genetic programming. *IEEE Transactions on Evolutionary Computation 1*, 2 (July 1997), 109–128.
4. Kruiskamp, M.W. Analog design automation using genetic algorithms and polytopes. Ph.D. dissertation. Dept. of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands, 1996.
5. Ochotta, E.S., Rutenbar, R.A., and Carley, L.R. Synthesis of high-performance analog circuits in ASTRX/OBLX. *IEEE Transactions on Computer-Aided Design 15* (1996), 273–294.
6. Sussman, G.J., and Stallman, R.M. Heuristic techniques in computer-aided circuit analysis. *IEEE Trans. Circuits and Systems 22* (1975).

Jason D. Lohn (jlohn@ptolemy.arc.nasa.gov) is a research scientist with the Caelum Research Corporation at NASA Ames Research Center in Moffett Field, CA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

© 1999 ACM 0002-0782/99/0400 $5.00

Planning to attend the Internet & Electronic Commerce '99 Conference in New York City on April 26-29?

Be sure to take advantage of the special 15% ACM member discount.

For more information or to register: www.iec-expo.com (mention code 143)

Just another great reason to be a member of the premier computing society.

acm